J. Symbolic Computation (1994) **17**, 457–472

Symbolic Local Refinement of Tetrahedral Grids

D. J. HEBERT

Department of Mathematics and Statistics

Pittsburgh, PA 15260

USA

(Received 23 July 1993)

A recent local grid refinement algorithm for simplicial grids is shown to be suitable for symbolic implementation in the 3-dimensional case. An addressing scheme stores all the geometric information about the tetrahedra in the refinement tree. Location of vertices and the addresses of physically nearest neighbors are computed by decoding the symbols of the simplex address. Bisection and face-compatible refinement of the simplex and its splitting neighbors are obtained by symbolic and logical operations on the leaves of the tree.

1. Introduction

Adaptive mesh refinement algorithms have a long history in the numerical analysis literature (cf. (Thompson, A.Warsi & Mastin [1985]), (Rheinboldt [1980]), (Baker [1989]), (Knupp & Steinberg [1993]), and references therein). They have become an essential part of modern computational tools, for example in finite element computations, in the approximation of differentiable manifolds, and in the related computer graphics of engineering design and geometric modeling (Babuska & Rheinboldt [1978]) (Rheinboldt [1988]) (McCormick [1989]) (Rossignac & Requicha [1991]). In such algorithms an initial coarse grid is locally refined automatically in regions where such features as densities, gradients, or curvatures require higher resolution. The special case of a grid of simplicial cells plays an important role both in analysis and in computing. The fact that a simplex has the minimum number of vertices and faces often means a smaller cost of communication with neighboring cells. Simplicial cells also allow the natural construction of finite element and spline approximations; for example, a function defined on the vertices of a simplicial grid has a piecewise affine extension which is affine on each simplex. The

0747–7171/94/050457 + 19 $08.00/0

study of simplicial mesh refinement also has a long history (cf. (Allgower & Georg [1979]) (Todd [1976]),). There have been many practical and theoretical problems to solve and progress is still being made (cf. (Bänsch [1991]), (Ferrucci & Paoluzzi [1991]), (Frey & Field [1991]), (Maubach [1994]), (Mitchell [1992]), (Rivara [1991])). In the present article we find that one very efficient simplicial refinement method may be made even more efficient by replacing numerical by symbolic computations.

The possibility of symbolic computation in this case is a consequence of the algebraic structure of a simplicial grid obtained by subdividing the unit cube. A grid is refined by bisecting simplices along their longest edge. A local symmetric structure repeats itself on a smaller scale at each third level of subdivision. Each simplex is represented as a unique expression in a simple algebra of translations, permutations, rotations, and scalings. This permits a labeling of the simplices and the computation of all properties of the grid by manipulation of the symbols in the labels.

1.1. Maubach's algorithm for simplicial refinement

In (Maubach [1994]), Maubach develops a simple bisection algorithm for local refinement of simplicial grids in n-dimensional space. The simplices are those obtained by successive bisecting of a chosen edge beginning in the congruence class of a reference simplex whose 3-dimensional version has vertices

$$(0,0,0),(1,0,0),(1,1,0),(1,1,1).$$

Some of the attractive features of the algorithm are:

1 – The edges to be bisected are determined without computation or global communication.
2 – Only n similarity classes are produced, one for each level (mod n) of refinement. As a consequence there is a fixed lower bound for vertex angles.
3 – The refinement is such that no incompatibilities are created., i.e., each face of a simplex created by the algorithm is shared by at most one other simplex.
4 – The algorithm is simple to implement and well-suited for parallel computing.

The present study, limited to the 3-dimensional version of the algorithm, introduces an addressing scheme which allows unique labeling of the simplices produced by Maubach's algorithm. The result is that the refinement algorithm, and the related facial neighbor algorithm may be implemented using only integer and logical computations. The label, and the position in the refinement tree structure, store all the information necessary to do numerical computations involving the simplices. Thus we see significant gains in both storage and computational efficiency over purely numerical methods. For example, the vertices of the simplex are computed from its label and the neighbors are computed by using only local labeling and inherent tree-structure information. There is no need to store large arrays of simplex neighbor links.

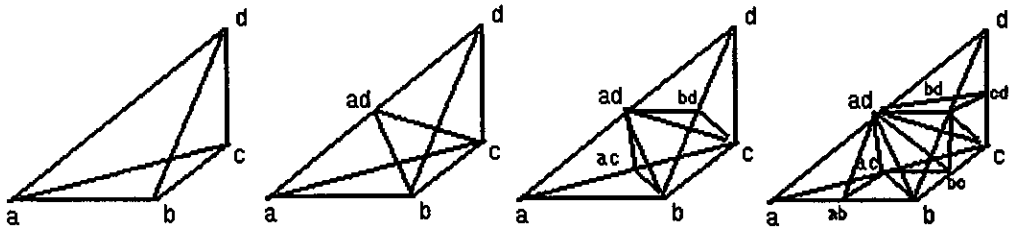


Figure 1. Repeated bisection of a tetrahedron (a, b, c, d)

1.2. Bisection and refinement in three dimensions

Maubach's simplex refinement algorithm is based on a bisection algorithm which produces a tree structure by successive bisection of simplices. In 3-dimensional space the algorithm starts with the specification of 6 congruent tetrahedra whose union is the unit cube $\mathbf{C} = [0, 1]^3$ and which share the diagonal joining the vertices $(0, 0, 0)$ and $(1, 1, 1)$. These 6 tetrahedra are called the simplices of level 0. The bisection algorithm in 3 dimensions works as follows: If a,b,c,d are the vertices of a simplex s of level 0 then we represent s as the list (a,b,c,d). The children of s are the level 1 simplices (a,b,c,ad) and (b,c,d,ad) where ad is a new vertex which bisects the edge joining a and d. The children of (a,b,c,ad) are (a,b,ac,ad) and (b,c,ac,ad). while those of (b,c,d,ad) are (b,c,bd,ad) and (c,d,bd,ad). These level 2 (mod 3) simplices split as follows:

(a,b,ac,ad) $\rightarrow$ (a, ab,ac,ad),(b,ab,ac,ad);
(b,c,ac,ad) $\rightarrow$ (b,bc,ac,ad),(c,bc,ac,ad);
(b,c,bd,ad) $\rightarrow$ (b,bc,bd,ad),(c,bc,bd,ad);
(c,d,bd,ad) $\rightarrow$ (c,cd,bd,ad),(d,cd,bd,ad).

The resulting level 3 simplices are congruent to half-size great grandparents. (Members of the same congruency class are obtained by rotations and reflections). Figure 1 shows the result of this algorithm applied to a particular simplex labeled (a, b, c, d).

The pattern repeats then as bisection continues. The result is a tree of dividing cells with a cube as the root, with 6 initial simplicial branches and with binary branching thereafter. Levels are assigned in the 6 subtrees of simplices. At level $3m + k$, $k = 0, 1, 2$, the simplices of these subtrees are congruent to the level k simplices scaled by a factor of $1/2^m$. Figure 2 shows the tetrahedra of levels 0,1,2,3.

A refinement algorithm typically selects simplices from a cellular grid for bisection according to some rule, for example whether a test function or its gradient exceeds a certain value in the simplex. To avoid incompatibilities, upon each bisection of a simplex it is necessary to bisect the neighbors which share the bisected faces. Therefore, an implementation must include some way of finding the appropriate neighbors to bisect. Furthermore, finite element applications will typically require information about all the facial neighbors. Our goal here is to describe versions of the bisection, refinement, and

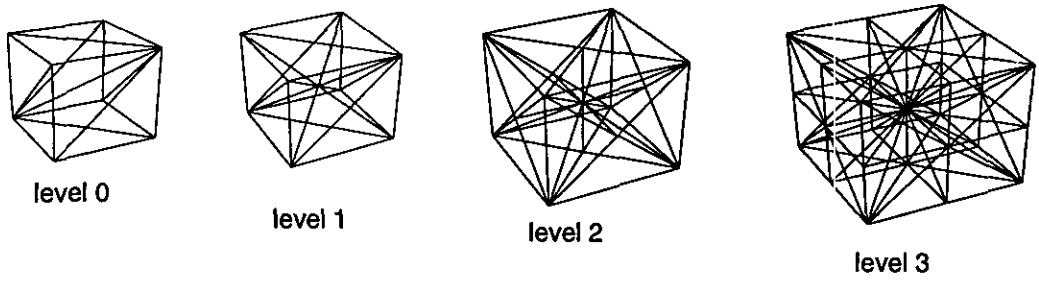


Figure 2. The tetrahedra of levels 0-3

neighbor-finding algorithms based on an efficient labeling scheme. A prototype implementation in the *Mathematica* programming language is used to test the algorithms.

2. Labeling of the simplex forest

In this section we shall see that the simplicial nodes of the tree which results from the bisection procedure may be uniquely labeled with a string of octal digits. The first 3 octal digits will determine the unique position of a simplex in an initial 3-level subtree which recurs at level $3m$ with $1/2^m$ scaling. The remaining digits locate the lattice origin of the simplex, a shared vertex for simplices of the scaled 3-level subtree. Computations are facilitated by representing a simplex by a matrix whose rows are the coordinate triples of the vertices.

We denote the unit basis vectors of $\mathbf{R}^n$ by $\mathbf{e}_i^n$, $i = 1, 2, \ldots, n$. As mentioned previously, we begin with reference simplex $\mathbf{s}_1$ whose vertices are 0, $\mathbf{e}_1^3$, $\mathbf{e}_1^3 + \mathbf{e}_2^3$, and $\mathbf{e}_1^3 + \mathbf{e}_2^3 + \mathbf{e}_3^3$. The vertices are the rows of the matrix

$$S_1 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

If I_{ijk} is the matrix whose rows are $\mathbf{e}_i^3$, $\mathbf{e}_j^3$, $\mathbf{e}_k^3$, then we define the permutation matrices Π_i by the formulae: $\Pi_1 = I_{123}$, $\Pi_2 = I_{213}$, $\Pi_3 = I_{231}$, $\Pi_4 = I_{321}$, $\Pi_5 = I_{312}$, $\Pi_6 = I_{132}$. The simplices $\mathbf{s}_i$ whose vertices are the rows of the matrix

$$S_i = S_1 \Pi_i$$

cover the unit cube $\mathbf{C} = [0, 1]^3$ and share the diagonal line segment joining $(0, 0, 0)$ and $(1, 1, 1)$ as a common edge.

If we denote each vertex of the unit cube $\mathbf{C}$ by the octal digit which is expressed in binary form by the coordinate triple of the vertex, then the level 0 simplices expressed as lists of coordinates are:

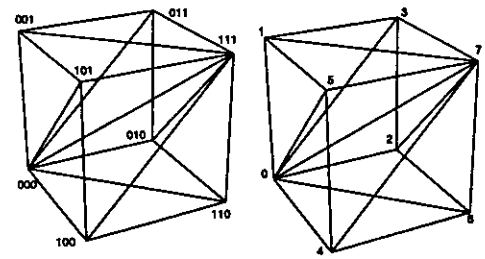


Figure 3. The set U with binary and octal digit labeling of the vertices of C

$$\begin{aligned} \mathbf{s}_1 &= (0,4,6,7), & \mathbf{s}_2 &= (0,2,6,7), \\ \mathbf{s}_3 &= (0,2,3,7), & \mathbf{s}_4 &= (0,1,3,7), \\ \mathbf{s}_5 &= (0,1,5\ 7), & \mathbf{s}_6 &= (0,4,5,7). \end{aligned}$$

(Figure 3 shows binary and octal labeling of the vertices of C). Let us denote the set of level 0 subsimplices by

$$U = \{\mathbf{s}_1, \ldots, \mathbf{s}_6\}.$$

It will be useful also to identify the reflections of the elements of U into the octants of $\mathbf{R}^3$. If j is an octal digit with binary representation

$$j = \sum_{k=0}^{2} b_k 2^k, \ b_k \in \{0,1\},$$

let $\sigma_k = 2b_k - 1$ so that $\sigma_k = 1$ when $b_k = 1$ and $\sigma_k = -1$ when $b_k = 0$, let

$$\rho_j = \mathrm{diag}[\sigma_1, \sigma_2, \sigma_3].$$

If we let $S_{ij} = S_1 \Pi_i \rho_j$, for $i = 1, \ldots, 6$ and $j = 0, \ldots, 7$ then the corresponding 48 simplices $\mathbf{s}_{ij}$ whose vertices are the rows of S_{ij} cover the cube $[-1,1]^3$. The diagonal triple of ρ_j represents the octant direction of the longest edge of $\mathbf{s}_{ij}$. Level 3 in figure 2 is an exact copy of the cube $[-1,1]^3$ sharing the 48 simplices $\mathbf{s}_{ij}$.

2.1. Matrix form of the bisection algorithm

Let $R_{\alpha\beta}$ be the matrix obtained from the 4×4 identity matrix by replacing row β with the average of row α and row β, and let $\Pi_{\alpha\beta}$ be the matrix obtained from the 4×4 identity matrix by right rotation of rows α through β so that multiplication on the left of a matrix A by $R_{\alpha\beta}$ or $\Pi_{\alpha\beta}$ performs the obvious row operations on A. . With these definitions we may inductively define the binary tree of descendants of S_{ij}:

$S_{ij}(1) = S_{ij}$ is at level 0.

If $S = S_{ij}(1, b_1, \ldots, b_q)$ is defined at level $q = 3m + l - 1$ where $m \geq 0$, $l = 1, 2, 3$, and b_α

is a binary digit for $\alpha = 1, \ldots, q$, then define the children S^0 and S^1 of S:

$$S^0 = S_{ij}(1, b_1, \ldots, b_q, 0) = \Pi_{l,4} R_{l,4} S,$$

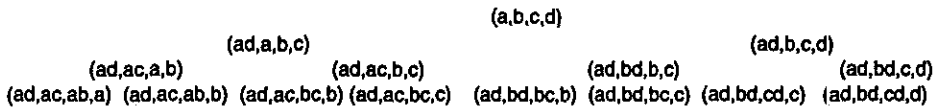
and

$$S^1 = S_{ij}(1, b_1, \ldots, b_q, 1) = R_{4,l} S.$$

The following lemma may be proved by direct observation:

LEMMA 2.1. *The simplices whose vertices are the rows of the matrices obtained above are exactly those obtained by Maubach's algorithm, except for the ordering of vertices.*

Indeed, if the vertices of a simplex s at level $3m$ are (a,b,c,d) then the bisection gives the following subtree:



Remarks and Corollaries

1 – The shortest edges of a level $3m$ tetrahedron are of length $1/2^m$.
2 – The longest edge $d - a$ of a level $3m$ tetrahedron points into the octant whose signs are given by $2^m(d - a) = (\sigma_1, \sigma_2, \sigma_3)$.
3 – The simplices of levels $3m + 1$, $3m + 2$, and $3m + 3$ share a common first vertex (ad in the above example) which is obtained by the formula

$$ad = a + \frac{1}{2}(d - a) = a + \frac{1}{2^{m+1}}(\sigma_1, \sigma_2, \sigma_3)$$

2.2. THE UNIT CUBE TETRAHEDRA s_{ijk} AND THE C-SIMPLICES

We define

$$S_{ijk} = S_{ij}(1, b_1, \ldots, b_q)$$

where

$$k = \sum_{\alpha=0}^{q} b_\alpha 2^{q-\alpha}, \; b_0 = 1.$$

S_{ijk} is called a matrix bisection descendant of S_{ij}. For $q = 0, 1, 2$ the subscripts i, j, k are octal digits; For $q = 3$ the subscript k is a hexadecimal digit. The 48 descendants at level 3 of the level 0 tetrahedra $\mathbf{s}_{i7}$ are half sized copies of the tetrahedra $\mathbf{s}_{ij}$, $i = 1, \ldots, 6$, $j = 0, \ldots, 7$, each copy having its first vertex at the point $(1/2)(1, 1, 1)$. Indeed, if we let

$\hat{1}$ be the 4×3 matrix of 1s and examine the descendants of S_{i7} we obtain the following lemma:

LEMMA 2.2. .

a *The descendants of S_{i7} of levels 1 and 2 are* $(1/2)(\hat{1} + S_{i7k})$ $k = 2, \ldots, 7$
b *At level 3 the descendants are*

$$S_{i7k} = \frac{1}{2}(\hat{1} + S_{pqr}),\ k = 8, \ldots, 15$$

where $S_{pqr} = S_{pqr}(i,k)$ is such that the triple p, q, r of octal digits is obtained from the following table:

i\k	8	9	10	11	12	13	14	15
1	400	441	540	561	640	661	160	171
2	500	521	420	461	320	361	260	271
3	600	621	120	131	220	231	330	371
4	100	111	610	631	510	531	430	471
5	200	211	310	351	410	451	550	571
6	300	341	240	251	140	151	650	671

The (i,k) entry in this table will be called level3Table(i,k)

Remarks concerning C-simplices

The simplices s_{i7} and their descendants will be called C-simplices and the corresponding matrices will be called C-simplex matrices.

1 – We note that the level 3 descendants of S_{i7} are of the form

$$\frac{1}{2}(S_{pqr} + \hat{1}\rho_7).$$

Since s_{ij} is a reflection of s_{i7}, i.e., $S_{ij} = S_{i7}\rho_j$, we may also observe that the level 3 descendants of S_{ij} are of the form

$$\begin{aligned}\frac{1}{2}(S_{pqr} + \hat{1}\rho_7)\rho_j &= \tfrac{1}{2}(S_1\Pi_p\rho_q\rho_j + \hat{1}\rho_j)\\ &= \tfrac{1}{2}(S_1\Pi_p\rho_{\hat{q}} + \hat{1}\rho_j)\\ &= \tfrac{1}{2}(S_{pqr} + \hat{1}\rho_j)\end{aligned}$$

where $\rho_{\hat{q}} = \rho_q \rho_j$.

2 – If

$$S = \frac{1}{2^m} S_{ijk} + \sum_{\alpha=1}^{m} \frac{1}{2^\alpha} \hat{1} \rho_{j_\alpha}$$

is a level $3m+2$ descendant of a level 0 C-simplex where $\rho_{j_m} = \rho_j$ represents the octant of S_{ijk} then the children of S are of the form

$$\begin{aligned} S^{k_1} &= \tfrac{1}{2^{m+1}} (S_{pqr}(i, k_1) + \hat{1}\rho_7)\rho_{j_m} + \sum_{\alpha=1}^{m} \frac{1}{2^\alpha} \hat{1} \rho_{j_\alpha} \\ &= \tfrac{1}{2^{m+1}} S_{p\hat{q}r} + \textstyle\sum_{\alpha=1}^{m+1} \frac{1}{2^\alpha} \hat{1} \rho_{j_\alpha} \end{aligned}$$

where $\rho_{\hat{q}} = \rho_q \rho_{j_m}$ and $j_{m+1} = \hat{q}$, $k_1 = 8, \ldots, 15$.

3 – Remark 2 gives the induction step in the proof of the following theorem.

THEOREM 2.1. *Each C-simplex matrix S of level $3m + l$, $l = 0, 1, 2$ obtained as a descendant of a level 0 C-simplex has a unique representation of the form*

$$S = \frac{1}{2^m} S_{ijk} + \sum_{\alpha=1}^{m} \frac{1}{2^\alpha} \hat{1} \rho_{j_\alpha}.$$

It follows that the C-simplices of level $3m + l$, $l = 0, 1, 2$ may be uniquely labeled by the strings of octal digits of the form $(i, j, k, j_1, \ldots, j_m)$ where $j_1 = 7$, and $i \in \{1, \ldots, 6\}$.

Remark: Since the first vertex of s_{ijk} is the origin, the vertex of the C-simplex s (whose matrix is S in the above theorem) is the first row of the matrix

$$\sum_{\alpha=1}^{m} \frac{1}{2^\alpha} \hat{1} \rho_{j_\alpha}$$

A symbolic bisection algorithm: The matrix bisection algorithm may seem simple enough, but the symbolic version uses even fewer computations. Indeed for the C-simplex $S_{ijkj_1 \ldots j_m}$ we see that

— for $k = 0, 1$, the children are $S_{ijk_1 j_1 \ldots j_m}$, where $k_1 = 2, 3$,
— for $k = 2, 3$, the children are $S_{ijk_2 j_1 \ldots j_m}$, where $k_2 = 4, 5, 6, 7$,
— for $k = 4, 5, 6, 7$ the children are $S_{p\hat{q}r j_1 \ldots j_m}$, where pqr =level3Table(i, k_1), $k_1 = 8, \ldots, 15$, and $\rho_{\hat{q}} = \rho_q \rho_{j_m}$.
— (The children of $S_{p\hat{q}r j_1 \ldots j_m}$ above are $S_{p\hat{q}\hat{r} j_1 \ldots j_m}$, where $\hat{r} = 2, 3$).

3. The neighbors of a C-simplex

If (v_1, v_2, v_3, v_4) is the vertex list for a tetrahedron, then the facial neighbors n_1, n_2, n_3, n_4 he face opposite the vertex of the same number. For example, n_1 shares the

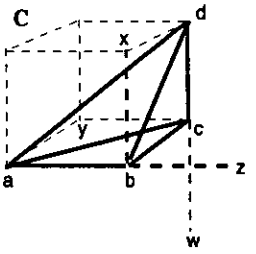


Figure 4.

face (v_2, v_3, v_4). Our goal in this section is to compute the symbols for the neighbors n_1, n_2, n_3, n_4 from the given symbol of a C-simplex s of level $3m+l$, $l = 0, 1, 2$. To do so requires information about the special case $l = 0$

Suppose that $s = (a, b, c, d)$ is of level $3m$ with neighbors $n_1 = (z, b, c, d)$, $n_2 = (a, y, c, d)$, $n_3 = (a, b, x, d)$, and $n_4 = (a, b, c, w)$, for example as in figure 4. Note that the vertices a, b, c, d are also vertices of a containing cube C_m with edge width $1/2^m$.

If $S = S_{pr\mathcal{d}j_1 \ldots j_m}$, we will call p the permutation number, r the reflection number, and d the descendant number, and $l = (j_1, \ldots, j_m)$ the lattice origin list of S and its corresponding C-simplex s. We denote by p_{ij}, r_{ij}, d_{ij}, l_{ij} the permutation number, rotation number, descendant number and lattice origin list of neighbor j of level i and observe the following:

1 – n_2 and n_3 are contained in the same cube C_m, the permutation numbers p_{02} and p_{03} differ from p by one, the reflection numbers are the same: $r = r_{02} = r_{03}$, and the descendant numbers d_{02}, and d_{03} may be obtained from p and r as the parity of the position (p, r) in the level3Table.

2 – Since the permutation number is determined by b and c without regard for signs, n_1 and n_4 have the same permutation numbers: $p_{01} = p_{04} = p$.

3 – The reflection number r_{01} of n_1 is determined by $d - z$. If $r = \sum b_i 2^i$ then $r_{01} = \sum b_i^1 2^i$ where b^1 is the Boolean complement of b. The reflection number of n_4 is determined by the vertex c. Indeed, if $\rho_j =$ diag $2^m(c-a)$ then the reflection number r_{04} is such that $\rho_{r_{04}} = \rho_r \rho_j$. The descendant number of n_1 and n_4 may be obtained as the parity of $(p01, r01)$ in the level3Table.

4 – n_2, n_3 and n_4 have the same lattice origin $l = l_{02} = l_{03} = l_{04}$.

5 – The lattice origin of n_1 is obtained by moving from the lattice origin of s a distance of $1/2^{m-1}$ in the coordinate direction $(2^m)(b-a)$ which may be determined from the permutation number according to the following table:

p	1	2	3	4	5	6
$2^m(b-a)$	3	5	5	6	6	3

The neighbor structure of descendants of the tetrahedron (a, b, c, d) may be studied more thoroughly by reference to the diagram of figure 5 which depicts C-simplices as nodes of a graph. The vertices of the central hexagon represent the level 0 tetrahedra which cover the cube C, labeled by the vertex triple (b, c, d) on one line and the three digits prd on a second line. On the first surrounding circle are located the level one children of these C-simplices labeled by the vertex triples (a, b, c) and (b, c, d). Here two neighbors occur on the two sides connected by the circular arc and a third neighbor, the child of a parent's neighbor is found on a connecting secant line. An additional neighbor is outside the cube and doesn't appear on the diagram. The next concentric circle, representing level 2, labeled by (ac, a, b), (ac, b, c), (bd, b, c), (bd, c, d), also shows neighbors to the left and right along the circular arc and along a secant line which connects to the child of a parent's neighbor. Again, an additional neighbor is outside the cube. The neighbors at level 3 are not shown, but they may be analysed in terms of the containing cube similar to the one at level 0. The diagram and its extensions to analogous ones for the neighboring cubes may be used to observe a number of facts about neighbors. Neighbor labels on the diagram are recognized by the sharing of 2 vertex labels.

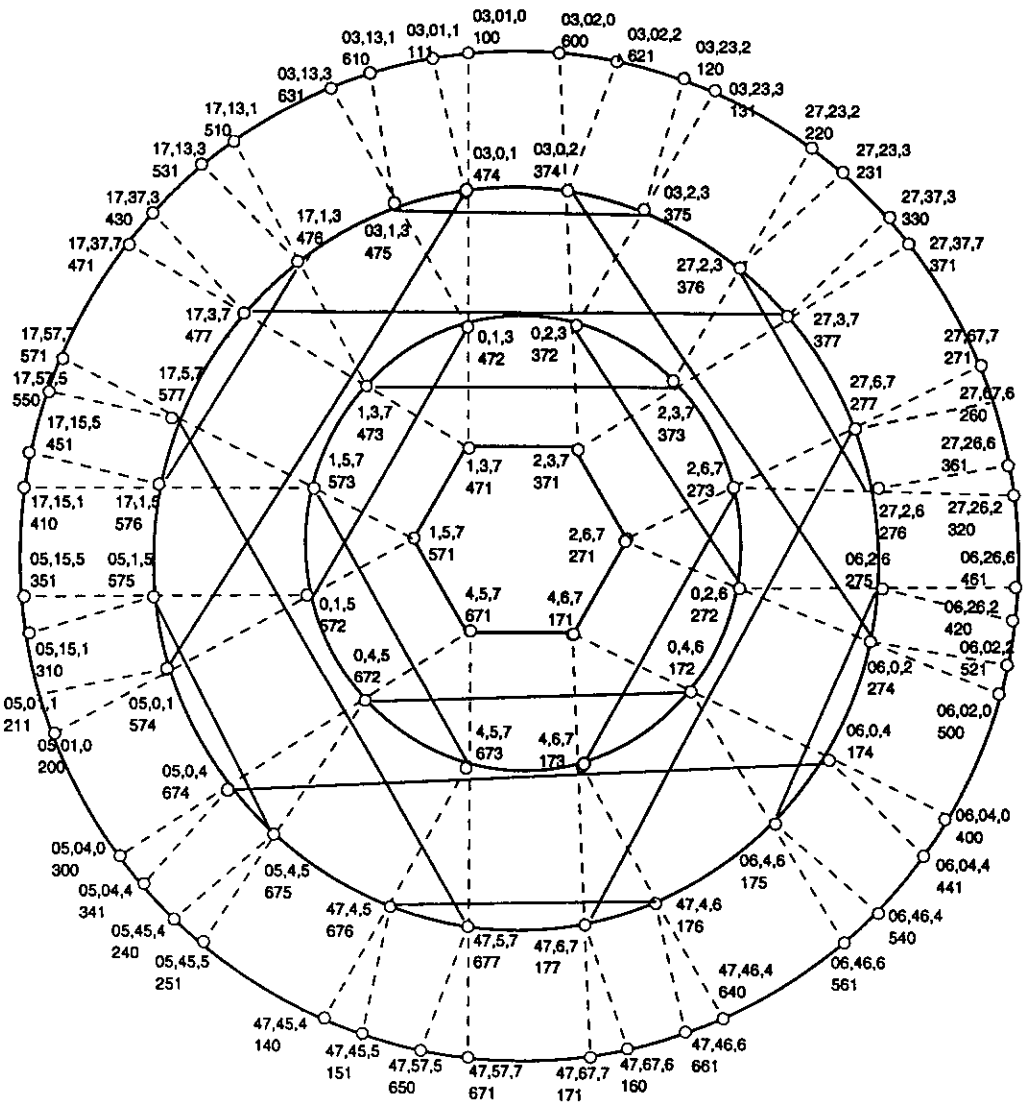
Observations

The numbers p_{0i}, r_{0i}, d_{0i}, and l_{0i} may also be computed as above for C-simplices of all levels and may be used as follows to compute the remaining numbers in the symbolic label.

At level 1 we have:

	p_{11}	$= p_{01}$	r_{11}	$= r_{01}$	d_{11}	$= d$	l_{11}	$= l_{01}$
$d = 2$	p_{12}	$= p$	r_{12}	$= r$	d_{12}	$= 5 - d$	l_{12}	$= l$
$d = 3$		$= p_{02}$				$= d$		
$d = 2$	p_{13}	$= p_{03}$	r_{13}	$= r$	d_{13}	$= d$	l_{13}	$= l$
$d = 3$		$= p_{02}$				$= d$		
$d = 2$	p_{14}	$= p$	r_{14}	$= r$	d_{14}	$= d$	l_{14}	$= l$
$d = 3$		$= p_{03}$				$= 5 - d$		

At level 2 we have:



neighbors of tetrahedra, levels 0-3

Figure 5.

p_{21}	$= p$	r_{21}	$= r_{01}$	d_{21}	$= d$	l_{21}	$= l_{01}$
p_{22}	$= p_{03}$, if $d = 4$ $= p_{02}$, if $d = 7$ $= p$, if $d = 5, 6$	r_{22}	$= r$	d_{22}	$= d$, if $d = 4$ $= d$, if $d = 7$ $= 11 - d$, if $d = 5, 6$	l_{22}	$= l$
p_{23}	$= 7 - p$, if $d = 7, p = 1, 6$ $= 3 - p$, if $d = 5, p = 1, 2$	r_{23}	$= r$	d_{23}	$= d + 1$, if d is even $= d$, if d is odd	l_{23}	$= l$
p_{24}	$= p$, if $d = 5, 7$ $= 7 - p$, if $d = 6$, $p = 1, 6$	r_{24}	$= r$	d_{24}	$= d - 1$, if d is odd $= d$, if d is even	l_{24}	$= l$

An algorithm for finding the level l neighbors of a C-simplex of level l is obtained by constructing a decision tree based on the above tables.

In addition to neighboring C-simplices at the same level for a C-simplex at level l there are possible neighbors at levels $l + 1$ and $l - 1$ but at no other levels. These may be observed by comparing vertices on figure 5.

If a C-simplex **s** shares a face with neighbor **t** at the same level and child i of **s** shares the same face with child j of **t** then $i = j$.

The edge which is bisected in the bisection algorithm is called the splitting edge. For the local refinement algorithm we need to identify the splitting neighbors, i.e. neighbors which share the splitting edge. These are observed on figure 5 to be the following:

— At level 0 the splitting neighbors are n_2 and n_3.

— At level 1 the splitting neighbors are n_1 and n_3.

— At level 2 the splitting neighbors are n_1 and n_2.

For a C-simplex of level l the splitting neighbors are either at level l or at level $l - 1$.

If the splitting neighbor $\mathbf{s}_1$ of a C-simplex **s** is at level $l - 1$ then the splitting edge of **s** differs from that of $\mathbf{s}_1$, but a child of $\mathbf{s}_1$ shares its splitting edge with **s**.

The C-simplices sharing a splitting edge form a cycle of neighbors at the same level (in the undirected graph whose nodes are the C-simplices and whose arcs are the neighbor pairs). At level $3m + 0$ there are 6 simplices in the cycle; at level $3m + 1$ there are 4, and at level $3m + 2$ there are 6.

4. Implementation of a symbolic local refinement operator

4.1. A SYMBOLIC LOCAL REFINEMENT ALGORITHM

Let T_0 be the tree whose set of nodes is $\{C\} \bigcup U$ such that C is the root node and U is the set of leaves. A tree T will be called *a C-simplex refinement tree* if

1 – T is obtained from T_0 by a finite number of applications of the matrix bisection operation,

2 – The mesh M of T is compatible, i.e. if faces f_1 and f_2, respectively, of simplices s_1 and s_2 have an interior point in common, then $f_1 = f_2$.

A C-simplex refinement tree is an example of a refinement tree in the terminology of Rheinboldt (Rheinboldt [1980]) , and the union of the collection of all C-simplex refinement trees is a refinement process. The set of leaves of a refinement tree T is called the mesh of T. Let $\mathcal{T}$ be the set of all C-simplex refinement trees. If $T \in \mathcal{T}$ and $s \in T$, define $\beta_S(T)$ to be the tree obtained from T by adjoining the children of s as computed by the matrix bisection algorithm. If the level of s is l then the splitting neighbor set $\mathcal{N}(s)$ will be the set of two level l neighbors of s which share the splitting edge of s.

We define a C-simplex refinement operator to be a mapping $\mathcal{R}$ whose domain and range are subsets of $\mathcal{T}$ such that for each $T \in \mathcal{T}$, $\mathcal{R}(T)$ is obtained from T by a finite number of applications of operators of the form β_S. We seek to implement a C-simplex refinement operator which acts only on the tree of symbolic labels of a C-simplex refinement tree and produces another tree of labels. The basic idea is the similar to Maubach's refinement algorithm (Maubach [1994]), but will take advantage of the observations of the previous section.

As mentioned earlier, a local refinement algorithm begins with a compatible simplex refinement tree and selects simplices from the mesh for bisection by a criterion such as the value of a test function defined on the mesh. To preserve compatibility the neighbors sharing the bisected edge must be bisected. However, the leaves of a compatible tree are not necessarily all at the same level. If a simplex s of level l is to be bisected and its neighbor s_1 in the given refinement tree is at level $l-1$ then a child of s_1 shares a splitting edge with s, hence the children of s_1 must be adjoined to the tree. But also all the children of splitting neighbors of s_1 must be adjoined. If some splitting neighbor of s_1 is at level $l-2$ then we must check its splitting neighbors a level $l-2$ to see whether they are in the tree. If not we continue until we reach a level at which the neighbors and the simplex are at the same level. Then we may apply the bisection algorithm to the simplex and its splitting neighbors to produce a compatible tree and continue the process of checking the splitting neighbors at the previous levels until we obtain a tree which contains all the splitting neighbors at level l for the simplex s. This procedure is made more precise in the following recursive definition and a theorem.

Given $T \in \mathcal{T}$ with mesh M, and $\mathbf{s} \in M$, we may define a refinement operator $\mathcal{R}_{\mathbf{S}}$ as follows:

$\mathcal{R}_{\mathbf{S}}(T)$:

foreach $\mathbf{s}_1 \in \mathcal{N}(\mathbf{s})$,

If $\mathbf{s}_1 \in M$, **then** $T \to \mathcal{R}_{\mathbf{S}_1}(T)$

else $T \to \mathcal{R}_{\hat{\mathbf{S}}_1}(T)$, where $\hat{\mathbf{s}}_1$ is the parent of $\mathbf{s}_1$.

$T \to \beta_{\mathbf{S}}(T)$,

THEOREM 4.1. *If* $T \in \mathcal{T}$ *then* $\mathcal{R}_{\mathbf{S}}(T) \in \mathcal{T}$

Suppose that ϕ is a $\{0,1\}$-valued function (a test function) whose domain includes the meshes of elements of $\mathcal{T}$. Suppose that $T \in \mathcal{T}$ and that M is the mesh of T. A refinement algorithm may then be expressed as follows:

For each $\mathbf{s} \in M$, if $\phi(\mathbf{s}) = 1$, $T \to \mathcal{R}_{\mathbf{S}}(T)$.

4.2. PROTOTYPES AND APPLICATIONS

The most efficient implementation of the symbolic refinement algorithm will depend upon the type of computer and the particular application. As a preliminary step, to test the algorithm, to gain experience, and to obtain results on small problems it is most convenient to implement a prototype in a symbolic algebra system . Indeed, for some applications the final implementation will be appropriate for such a system. The main data structures are lists, tables, and trees and the main functions are facilitated by the use of list or string processing capabilities. The local refinement algorithm requires list membership testing and a few simple functions which make decisions according to the tables given in previous sections. For example, to compute $\mathcal{N}(\mathbf{s})$ we use the tables of section 3 to construct a function which produces the neighbors of a C-simplex from its symbolic label; to compute $\beta_{\mathbf{S}}(T)$ we must implement the symbolic bisection algorithm given in section 2.2; and finally, to compute $\hat{\mathbf{s}}_1$ we need a function to compute the parent of a given C-simplex.

The only numerical calculation is that of the lattice origin for neighbors external to the local containing cube of a given simplex. This calculation may be performed with only integer operations since it involves binary and octal digits of a finite expansion. The vertices of the simplices which may be needed to compute the values of the test function ϕ, are computed simply from the symbolic label, and may be expressed as binary points

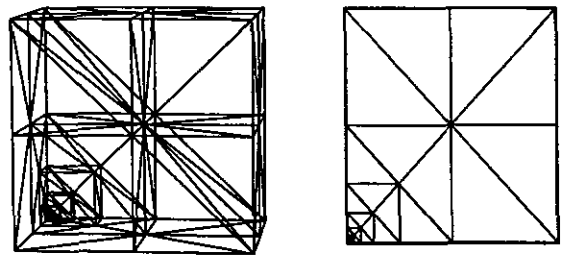


Figure 6. Refinement around a vertex of C

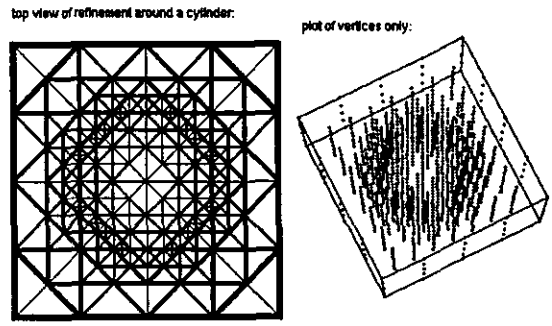


Figure 7. Refinement on the surface of a cylinder

of the unit cube, i.e. as triples of numbers of the form

$$\sum_{k=1}^{m} \frac{b_k}{2^k} \text{ where } b_k \text{ is a binary digit.}$$

A prototype written in the *Mathematica* programming language was used to study the neighbor structure and to generate the tables and pictures of this paper. Some sample results are shown in figures 6 and 7. Figure 6 shows two views of the result of refining those simplices which contain a given vertex of the cube C. Figure 7 shows two views of the result of refining those simplices which contain the surface of a cylinder. It has been noted in many such examples that Maubach's algorithm has the property that the compatible refinement near a sharp interface does not spread widely in 3-dimensional space. A *Mathematica* Notebook (Hebert [1992]) and an implementation package containing the details and examples are available via anonymous ftp at internet addresses obtainable from the author.

A symbolic local refinement algorithm may provide the possibility for efficient non-numerical approaches to manifold approximation and geometric modeling within symbolic algebra systems. For simulation programs with large-scale adaptive grid changes it may be appropriate to implement the algorithm in lower level languages where using only integer and logical operations. For some applications such as optimization problems and higher dimensional manifold approximation it may be of interest to develop a similar algorithm for simplicial grids in n dimensions. Extensions of the symbolic labeling to n

dimensions is naturally accomplished by using 2^n-ary digits instead of octal digits. The bisection algorithm is also extended in a straight-forward manner: the n-permutations form level 0 and the recurring binary tree has n levels. The neighbor-finding algorithm in two dimensions is trivial, but for general $n > 3$ it is a definite challenge.

References

E. Allgower & K. Georg [1979], "Generation of Triangulations by Reflections ," *Utilitas Mathematica* 16, 123–129.

I. Babuska & W. Rheinboldt [1978], "Error estimates for adaptive finite element computations ," *SIAM J. Numer. Anal.* 15, 736–754.

T. J. Baker [1989], "Developments and trends in three-dimensional mesh generation," *Applied Numerical Mathematics* 5, 275–304.

E. Bänsch [1991], "Local mesh refinement in 2 and 3 dimensions," *Impact of Computing in Science and Engineering* 3, 181–191.

V. Ferrucci & A. Paoluzzi [1991], "Extrusion and boundary evaluation for multidimensional polyhedra ," *Computer Aided Design* 23, 40–50.

W. H. Frey & D. A. Field [1991], "Mesh relaxation: a new technique for improving triangulations. ," *Int. J. Num. Meth. in Engineering.* 31, 1121–1133.

D. J. Hebert [1992], *Symbolic Simplex Refinement in 3D, a study of Maubach's Algorithm*, Technical Report ICMA-92-176 , Department of Mathematics and Statistics, University of Pittsburgh, Nov. 1992.

P. Knupp & S. Steinberg [1993], *Fundamentals of Grid Generation*, CRC Press, Boca Raton, Florida, ISBN 0-8493-8987-9.

J. M. Maubach [1994], *Local bisection refinement for n-simplicial grids generated by reflections.*, (to appear in SIAM J. Sci. Stat. Comput.).

Stephen F. McCormick [1989], *Multilevel Adaptive Methods for Partial Differential Equations*, SIAM, Philadelphia.

William F. Mitchell [1992], "Optimal Multilevel Iterative Methods for Adaptive Grids," *SIAM J. Sci. Stat. Comput.* 13, 146–167.

Werner C. Rheinboldt [1980], "On a Theory of Mesh-Refinement Processes," *SIAM J. Numer. Anal.* 17 , 766–778.

Werner C. Rheinboldt [1988], "On the Computation of Multi-Dimensional Solution Manifolds of Parametrized Equations," *Numerische Mathematik* 53 , 165–181.

M. C. Rivara [1991], "Local modification of meshes for adaptive and/or multigrid finite element methods," *J. Comp. Appl. Math.* 36, 79–89.

J. R. Rossignac & A.A.G. Requicha [1991], "Constructive Non-Regularized Geometry," *Computer Aided Design* 23, 21–32.

J. F. Thompson, Z. U. A.Warsi & C. W. Mastin [1985], *Numerical Grid Generation*, Elsevier, New York.

M. J. Todd [1976], *The computation of Fixed Points and Applications*, Springer, New York, Berlin, Heidelberg.